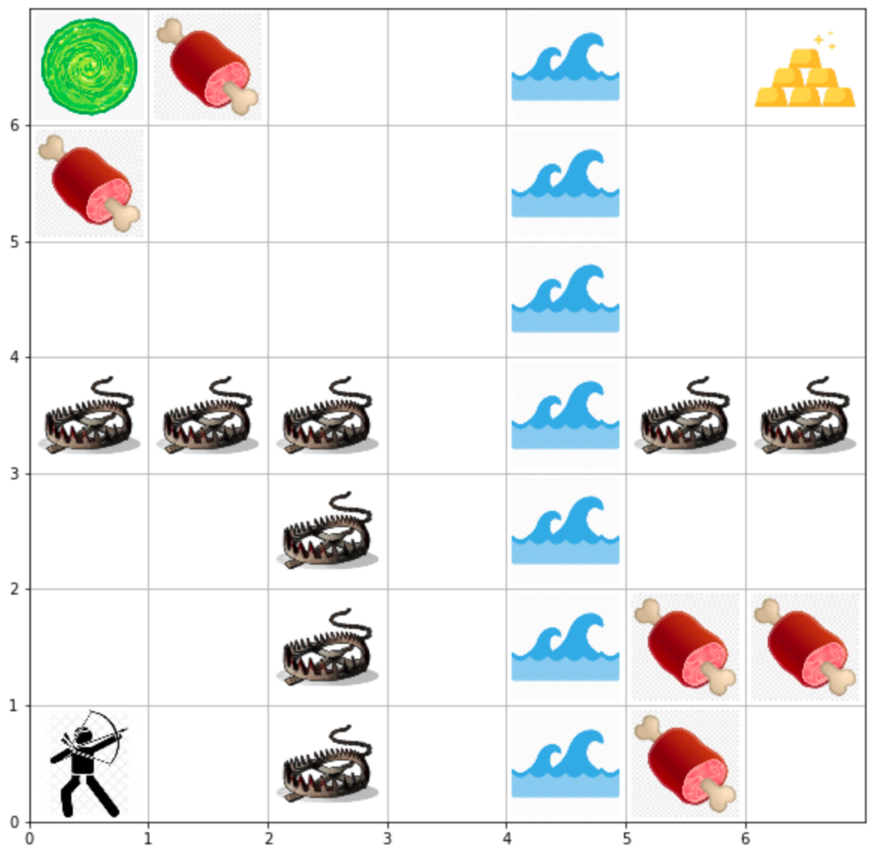
Environment Overview

1. **Describe the deterministic and stochastic environments, which were defined (set of actions/states/rewards, main objective, etc).**

2. **Provide visualizations of your environments.**

The environment represents a grid world game called GoldHunter. In this game a hunter starts the game from the cell (0, 0) and should try to maximize his score after finishing the game. The game generally consists of multiple objects which each of them have different properties and behaviours. An overview of the map of the game is shown in the below image:



First, I start by defining the environment by its fundamental components and then we'll dive into the dynamics of the game.

States = {49 positions on the grid world}
Actions = {UP, RIGHT, DOWN, LEFT, LOG_UP, LOG_RIGHT, LOG_DOWN, LOG_LEFT}
Rewards = { -5, -3, -1, +5, +10}

Objects

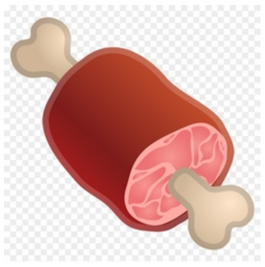
Hunter

Hunter is our agent and is trying to reach the gold, while maintaining his power. He can maintain his power **by avoiding traps, reaching to gold faster or eating food**

Gold

Obviously this is our goal. By reaching the cell containing gold the episode terminates and agent receive a reward of 10

Food

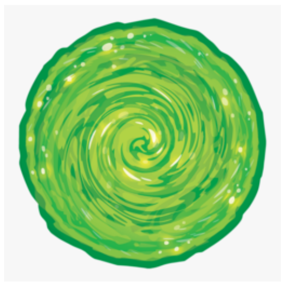


Food is beneficial for the hunter because it keeps him from getting hungry :(
Although, hunter can find the gold and finish the game, but, collecting food before that maximizes his reward.

Stream

Streams can be both dangerous and beneficial! When the agent falls into one, the stream takes him to the last place it goes. If the stream ends at the border of grid world the agent falls on the left or right of the stream!

Teleport



Teleport is a non deterministic element of this world. It connects the lower right to the higher left; however, it is non deterministic and it might not get activated when the hunter first goes inside it.

Trap

Traps are tricky creatures! When the hunter falls into one, he gets hurt and gets a reward of -5.

Log

Logs are objects that the hunter can craft around him. Hunter has 4 other actions other than his movement actions to each direction. These actions craft a log around him in each direction and lets him pass the object ahead of him without being affected by its properties. For example, he can build a log on the stream or a trap and pass on it. However, **crafting is costly and makes him tired**, so he'll receive a reward of -3 when he builds the log.

Dynamics of the environment

3. How did you define the stochastic environment?

Deterministic environment

Agent starts at (0, 0) and tries to get the gold by going to (6, 6). In this journey, the agent might fall into traps which hurt him and affect him by negative reward. He can also craft a log on top of the trap, before falling into it.
Also there is the possibility of falling into the stream which as described above, takes him to the end of stream.

Stochastic environment

To add stochasticity to the environment, I added a teleport channel, which can help the agent teleport to the other corner of the map where he can access food! The teleport lets the hunter move to the other corner with probability of P.

There is also the probability of finding food in 2 opposite corners of the map.

4. What is the difference between the deterministic and stochastic environments?

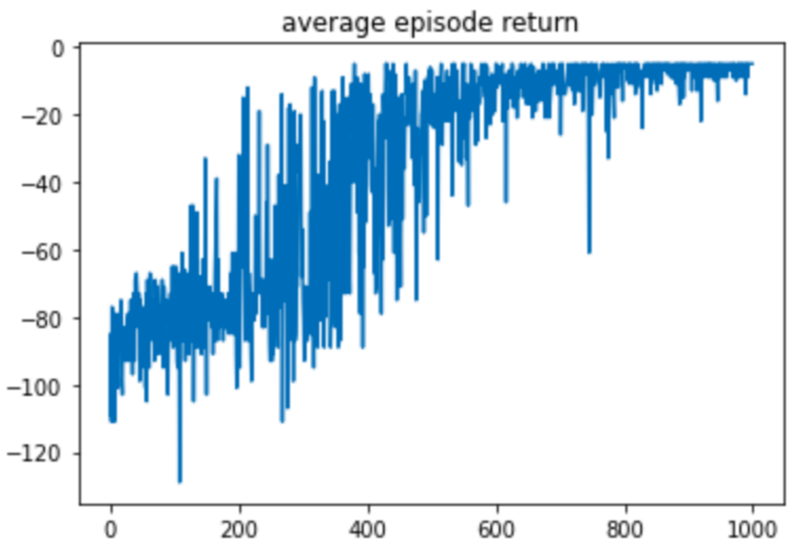
In deterministic environment, given that you are in state S and you perform action a you will end up in a specific state S' and receive reward r. Although, In stochastic environments, there is a probability for this transaction and you might end up in a different state and receive a different reward as well.
In GoldHunt environment as an example the following stochastic behaviors exist:

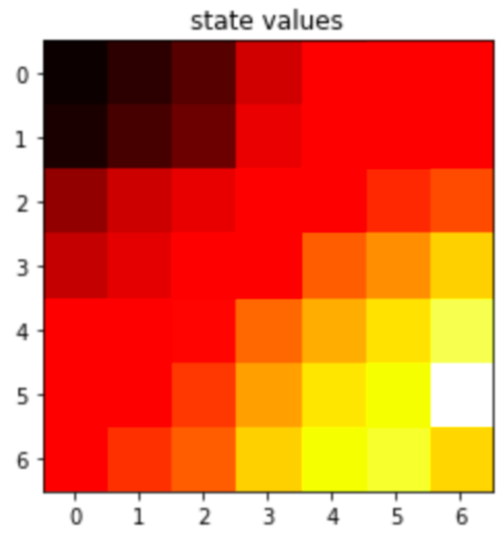
$$P([0,6],\ +5 \mid [5,0],\ RIGHT) = P$$
$$P([6,0],\ -1 \mid [5,0],\ RIGHT) = 1-P$$
$$P([0,6],\ +5 \mid [6,1], DOWN) = P$$
$$P([6,0],\ -1 \mid [5,0],\ DOWN) = 1-P$$

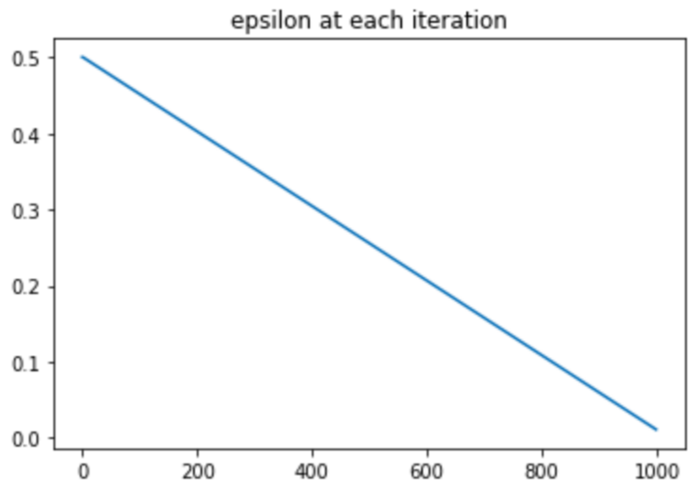
Also there is a non-determinism in this environment which can lead the environment to become a **non markovian decision process**, and that is **the food element.** The reason is that after eating the food, the transition probabilities around food elements change. That is a challenging element and makes it harder for the agent to work, but I will consider it in this phase.

Solutions

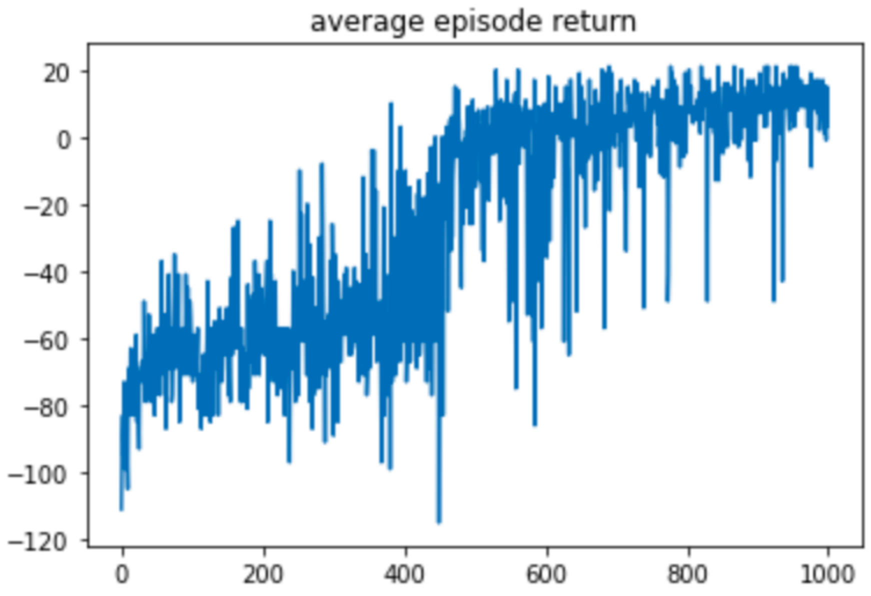
Applying Q-learning to solve the deterministic environment defined in Part 1. Plots should include epsilon decay and total reward per episode

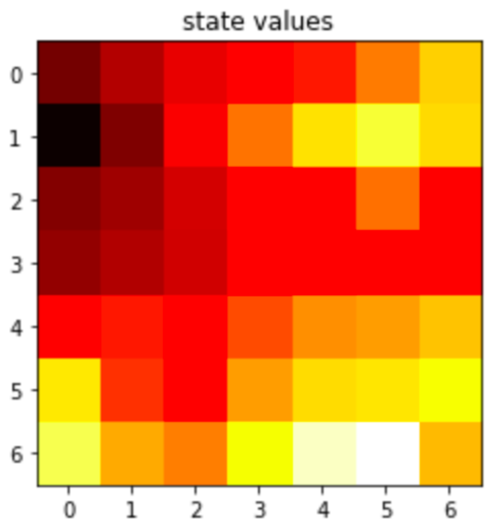


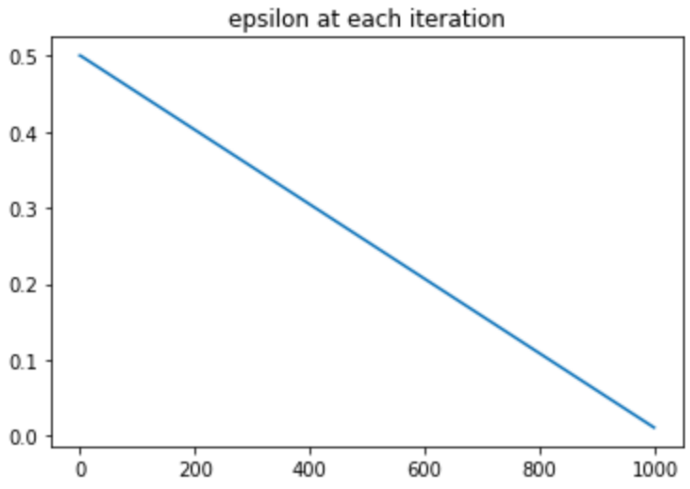




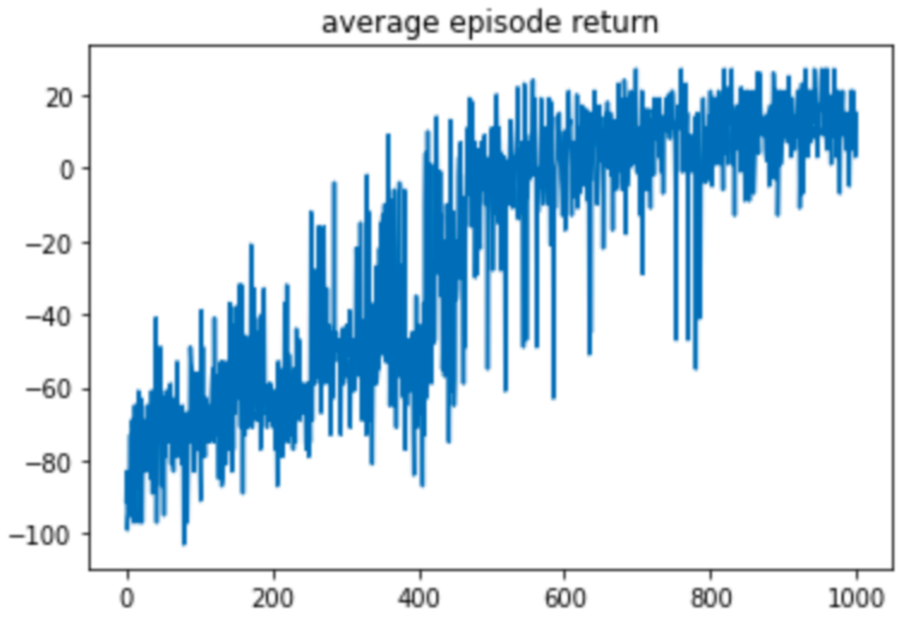
Applying Q-learning to solve the stochastic environment defined in Part 1. Plots should include epsilon decay and total reward per episode.

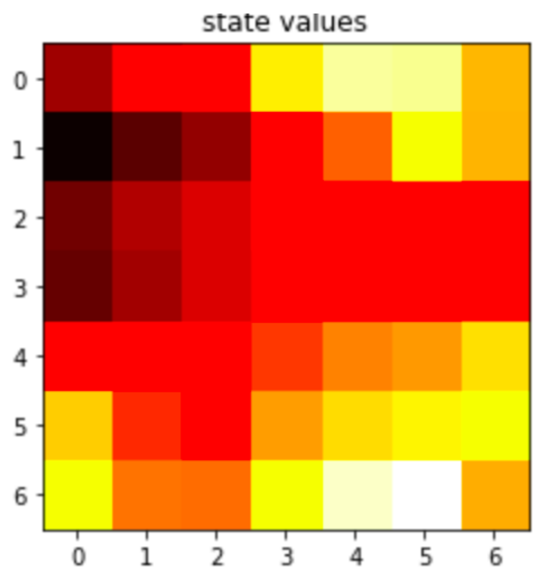


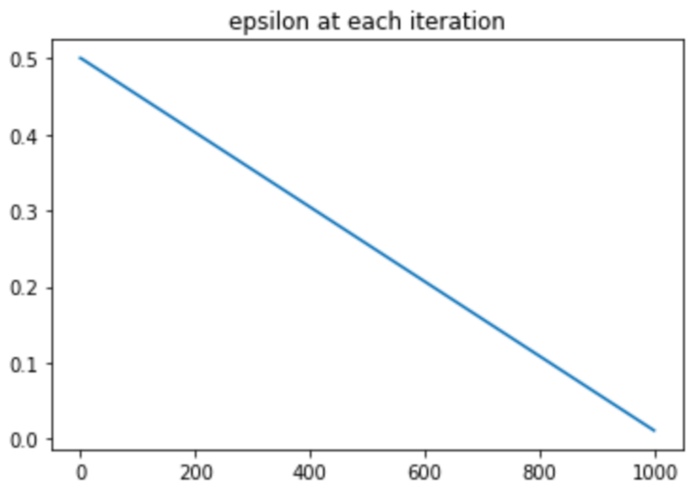




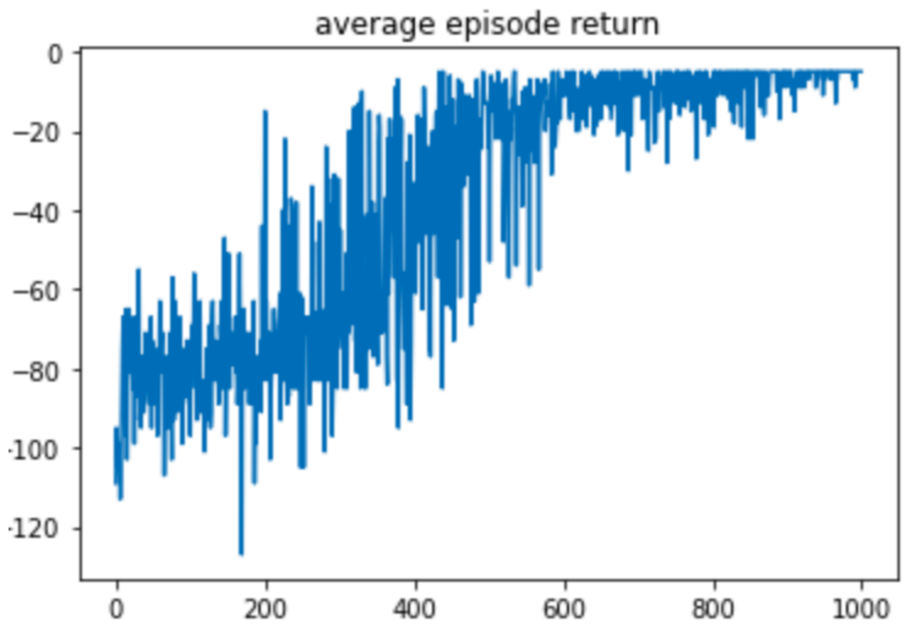
Applying any other algorithm of your choice to solve the deterministic environment defined in Part 1. Plots should include total reward per episode.

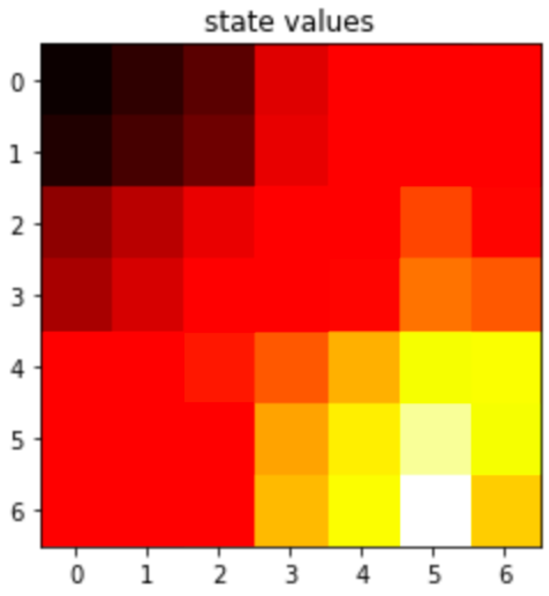


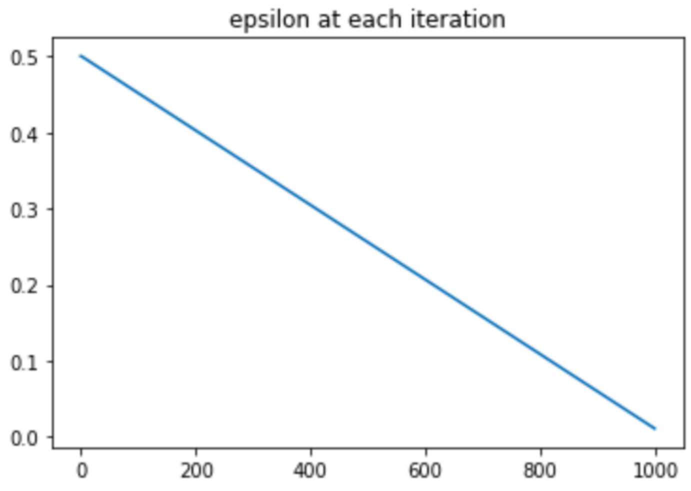




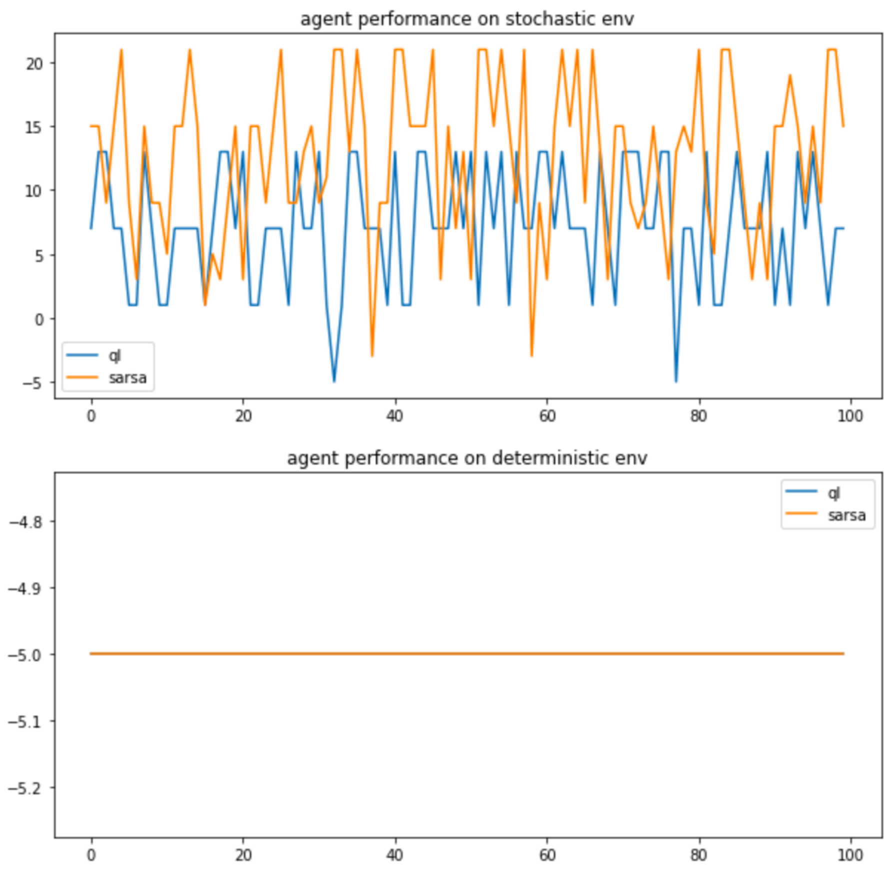
Applying any other algorithm of your choice to solve the stochastic environment defined in Part 1. Plots should include total reward per episode.







Provide the evaluation results. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode



Compare the performance of both algorithms on the same deterministic environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.

As we see in the above graph both Qlearning and Sarsa agents converge to the optimum reward and both result in the reward of -5.

Compare how both algorithms perform in the same stochastic environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.

Again, based on the above graph, sarsa is achieving a better performance compared to qlearning. Although, under different configurations and hyper parameters, both algorithms can achieve around the same good returns. But in this experiment, sarsa wins!

Briefly explain the tabular methods, including Q-learning, that were used to solve the problems. Provide their update functions and key features.

In general, as their name states, tabular methods provide a table of either action values or state values to solve the problem. To do this, they mostly follow multiple iterations of policy evaluation and policy improvement; however, the procedure is different in each algorithm.

For instance, in Q-learning, at each iteration the agent completes an episode and by iterating through each state of the episode we try to update Q-values. The update equation is provided below:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

Afterwards, the policy gets improved and to also keep exploring the environment performs a combination of random and greedy actions by following the epsilon greedy algorithm.
In the SARSA method, the only difference is that in the target term we won't choose the action with best q-value for bootstrapping, instead we choose the action that policy prefers according to the epsilon greedy algorithm. Therefore, the update function would be:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big]$$

In practice, I observed that when the **Q-learning agent plays in the environment, he takes food from the upper left and then builds a bridge on top of the river and finds gold. But the SARSA agent tries to get into the teleport after collecting food from the upper left and then he collects food from lower right and finishes the game by collecting gold**.

You can reproduce this result by loading agents' q-values and playing the game inside notebook.